Learning to Recommend Links using Graph Structure and Node Content

Antonino Freno[2] **Gemma C Garriga**[2] **Mikaela Keller**[1]
[2] INRIA Lille Nord Europe, France
first.last@inria.fr
[1] Université de Lille 3, France
first.last@inria.fr

Abstract

The link prediction problem for graphs is a binary classification task that estimates the presence or absence of a link between two nodes in the graph. Links absent from the training set, however, cannot be directly considered as the negative examples since they might be present links at test time. Finding a hard decision boundary for link prediction is thus unnatural. This paper formalizes the link prediction problem from the flexible perspective of preference learning: the goal is to learn a preference score between any two nodes—either observed in the network at training time or to appear only later in the test—by using the feature vectors of the nodes and the structure of the graph as side information. Our assumption is that the observed edges, and in general, shortest paths between nodes in the graph, can reinforce an existing similarity between the nodes feature vectors. We propose a model implemented by a simple neural network architecture and an objective function that can be optimized by stochastic gradient descent over appropriate triplets of nodes in the graph. Our first preliminary experiments in small undirected graphs show that our learning algorithm outperforms baselines in real networks and is able to learn the correct distance function in synthetic networks.

1 Introduction

Link prediction is a key problem in graph mining that has become fundamental to applications in recommendation systems, social networks, market analysis, and so on.

The problem of predicting links in a social network is a binary classification task that estimates the presence or absence of a link between two nodes in the graph. Formally it can be casted as follows: given an undirected graph at time t, namely $G_t = (V, E)$ with nodes V and edges $E \subseteq V \times V$, predict the presence of new edges in the evolved graph at time $t + 1$, namely $G_{t+1} = (V', E')$ for $V' \supseteq V$ and $E' \supseteq E$. Two types of settings for this problem have been identified by the machine learning community ([15]). In the *structural* setting, graphs at times t and $t + 1$ have the same fixed set of nodes and only new edges are expected to appear at time $t + 1$; in the *temporal* setting, the initial graph is expected to evolve more freely, by growing not only with new edges but also with new nodes and their associated new edges. In this paper we focus on the prediction and recommendation of new edges in both settings.

Challenges for the general link prediction problem have been clearly outlined in the recent literature [14, 15]. A first challenge concerns the extreme sparsity exhibited by the network datasets: the number of edges known to be present at time t are much less than the number of absent edges; also, the number of edges to be predicted at time $t + 1$ will be most probably small. Several drawbacks arise from this challege to the solution based on a binary classification. The first is the unreliability of the absent egdes as natural negatives examples; indeed in many cases the absent edges are simply unknowns to be predicted. A second drawback is that finding a hard boundary between the predicted classes in the test set is rather unnatural when so few edges will have to be predicted.

Previous literature deals with these drawbacks with different strategies. For example, factorizations of the adjacency matrix of the graph offers an approach to the link prediction problem in a matrix completion setting [1, 18, 15].

Such solutions are however limited in that a fixed set of nodes has to be considered at both time t and time $t+1$, hence it cannot deal with the real temporal prediction of network evolution that we will be also considering here.

Our approach deals with these issues by redefining the link prediction task as a *link preference* problem. Reasoning with preferences offers the flexibility of making predictions based on recommendations which could be violated. Our goal will be to learn a preference score $f(x, x')$ for every node x to any other node x'—either currently present in the network or to appear in the future—indicating their (mutual) affinity given the current network configuration. We will then use the scores $f(x, \cdot)$ to rank the possible edges that might appear connected to x at time $t+1$.

In order to learn preference scores for all nodes, even those not observed at current time t, we use both current structure observed in G_t and content information for the nodes. We suppose that every node x (present or future) has an associated feature vector $x \in \mathbb{R}^d$. Our algorithm will learn the preference score f to be a similarity function between pairs of feature vectors using the structure of graph G_t as a feedback. Indeed, it is challenging for link prediction to combine both content of the nodes and structure of the graph in a reasonable way: while structure plays an important role to predict within-community links, feature vectors are necessary to recommend links connecting nodes with strong affinities, which might be within- but also outside-community links. Naturally, these type of links crossing the community borders cannot be predicted by using only the structure of the graph if communities are disconnected at time t. Our assumption here is that it is possible to find an embedding of the nodes that would both respect the proximity constraints encoded in the observed edges, and in general, shortest paths between nodes in the graph, as well as the similarities in the feature space.

We propose an optimization function that properly models the singularities of this link preference problem. To optimize our objective function and learn the final preference score function f, we use neural networks, which will allow us to perform online optimization by sampling some set of appropriate triplets of nodes in the graph. An added benefit of such approach is that learning the score function can therefore be performed online, thus rendering it scalable in the case of very large networks.

The problem of learning a preference score for links in a graph as explained here can be seen as a special case of the *instance ranking* setting from preference learning [5, 6]. As in instance ranking, our goal is to produce a ranking function $f(x, \cdot)$ that will order a new set of instances for a given node x; however, our instances are not independent anymore, and therefore, our final f should take into account on the connectivity structure of the instances given by the graph too.

We present in this paper initial experiments that compare our approach on undirected binary graphs to several baselines approaches predicting links using exclusively the structure of the graph or a distance metric between feature vectors. Motivated by similar preference learning problems and because it is more sensitive in imbalanced datasets, our comparison for the link prediction scenario will be based on area under the ROC curve (AUC). Our first results show that our algorithm is able to learn the correct distance function in synthetic networks and outperform baselines in real networks.

2 Related Work

With the increased popularity of online social networks and also with the advent of new network completion problems in bioinformatics, the task of predicting links (mostly in undirected graphs) has been extensively studied in the recent literature. In general, popular graph-based proximity measures like personalized page-rank, Adamic/Adar or commute times are used for link prediction on graphs. The experimental setup performed in [13] showed that basic heuristics using an ensemble of short paths between two nodes, such as Katz measure [8], often perform the best from these baselines. These models are considered to be *unsupervised*; they do not involve any learning and their scores are predefined given the specific structure of the input graph and they have been analyzed theoretically in [16].

Supervised models on the other hand, aim at performing link prediction by viewing it as a binary classification task. A popular formalization has been to express the prediction of links as a matrix completion problem, solved via matrix factorizations of the adjacency matrix of the graph. In [1] the goal is simply to factorize the adjacency matrix of the graph. Works such as [18, 15] show how edge or node features can be incorporated into the factorization formulation. Also the work in [9] proposes a matrix factorization approach based on kronecker graph operations. These approaches have become very popular as a natural extension from the collaborative filtering problem in recommender systems, which exhibits similar challenges to link prediction [10]

Another different supervised approach to link prediction is [11], where the goal is to learn a transformation on the graph's algebraic spectrum which will serve as prediction of the linking potential scores. There are also approaches, for example [14], based on learning the right random walk in the graph so that a measure similar to page-rank can be computed.

As we do in this paper, all these aforementioned supervised approaches learn some sort of scores for each entry in the graph G_t in order to "complete" the graph at time G_{t+1} or propose ranks of affinity between nodes. The natural advantage of our approach over the others is that, by learning a supervised preference score, we are able to classify (i.e., to rank) the links of nodes that have never been seen before in the training graph G_t. Indeed, proposed solutions mentioned here are basically for structural prediction.

From the perspective of learning a distance metric, our problem formalization can be considered similar to [17]. However, our feature vectors are not independent; we are dealing here with a graph that estabilishes dependencies between feature vectors at time t.

Finally, it is worth noting as related work that the problem of learning a preference score for links in a graph has similarities with the *instance ranking* setting from preference learning [5]. As in instance ranking, our goal is to learn a ranking function f that will order a new set of instances. Again, in instance learning instances are considered to be independent one of the other; in our case, instances (nodes) are constrained by the connectivity structure observed in the input graph.

3 Our Approach

Let us assume that we know, up to a certain point in time, the interactions that occur between a group of entities. Let us also assume that we have a set of d real attributes describing those entities. To simplify notations in the remainder of the text, we will use x_i both to refer to an entity i and to its associated vector of attributes. As stated above, we would like to learn a ranking function allowing to order every pair of observed entities, as well as new upcoming entities, with respect to any other pair of entities, according to their respective potential for interaction or mutual affinity. We first make the hypothesis that the potential for interaction has some manifestation in the feature space, that is that two pairs of entities which have similar combined features would have similar affinities.

To put it more formally, let us assume that we are given a graph $G_t = (V_t, E_t)$ summarizing the interactions E_t collected at time t between a set V_t of n entities. We are also given, for each node in the graph G_t, a vector of attributes in $\mathbb{R}^d$. We would like to learn a function

$$f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

such that for all triplets of entities $x, x_+, x_- \in V_t$ with, $(x, x_+) \in E_t$ and $(x, x_-) \notin E_t$:

$$f(x, x_+) > f(x, x_-). \tag{1}$$

In the same vein, we can also express constraints similar, to a certain extent, to the one of equation (1) for triplets x, x_+, x_-, where x is connected to x_+ through a path of length at most k, and there is not such a short path to x_-. At time $t + 1$, there will be newly created links, between nodes already present at time t as well as links involving new nodes, in the graph $G_{t+1} = (V_{t+1}, E_{t+1})$, where $V_t \subseteq V_{t+1}$ and $E_t \subseteq E_{t+1}$. New nodes will also be provided with a a vector of attributes in $\mathbb{R}^d$. The assumption is that those new nodes will have features drawn from the same distribution as the nodes in V_t and that the linking process is preserved from one time step to the other so that those new links would have a high score f in the ranking of pairs of nodes.

For the sake of simplicity we restrict ourselves here to undirected graphs, that is to interactions that are symmetric. However this model could be easily generalized to directed graphs. In the simpler

symmetric case, equation 1 can be reformulated as:

$$\psi(x, x_+, x_-) = \phi(x) \cdot \phi(x_+) - \phi(x) \cdot \phi(x_-) > 0. \tag{2}$$

where ϕ is the embedding of the nodes feature vector into a new representation, and ψ is the relative comparison of the similarity (dot product) of the nodes of the triplet in that new representation. We expect this embedding to take both into account smoothness constraints with respect to the feature space and preferences of relative proximity from the structure of the graph[1].

Let T_1 be the set of valid triplets (x, x_+, x_-), such that x is connected to x_+ but not to x_-, that is $(x, x_+) \in E_t$ and $(x, x_-) \notin E_t$. Generalizing, let T_k be the set of triplets (x, x_+, x_-), such that the shortest path connecting x to x_+ is of length k and there is no path connecting x to x_- or it is of length more than k. Our cost function to optimize reads,

$$\mathcal{C} = \sum_{(x,x_+,x_-)\in T_1} \text{loss}(\psi(x, x_+, x_-)) + \sum_{k=2}^{K} \alpha_k \sum_{(x,x_+,x_-)\in T_k} \text{loss}(\psi(x, x_+, x_-)) \tag{3}$$

with $1 < \alpha_1 < \cdots < \alpha_K$, so that more importance is given to triplets involving direct interactions. Cost functions similar to the first element of the sum in equation (3) have been called ranking criterion with proximity constraints by [17, 3, 7]. In our case a set of k terms with decreasing importance are added to the sum that exploits longer path proximity constraint given by the graph structure. A function such as $\psi(x, x_+, x_-)$ can be learned using a model similar to the *Siamese* neural network proposed by [2] and more recently explored by [4]. In our case, ϕ is a perceptron with a non-linear activation function, replicated three times for x, x_+ and x_-, and learned by stochastic gradient descent optimization of $\mathcal{C}$, defined with a hinge loss, chosen for its known good generalization properties as follows:

$$\text{loss}(\psi(x, x_+, x_-)) = max(0, \psi(x, x_+, x_-) + 1)$$

During the online training process, instead of explicitely choosing the α_k weigths, we randomly sample triplets from $T_k,\ 1 \leq k \leq K$ with a bias inversely proportional to k.

At test time to every pair of nodes with feature vectors x and x' we are able to associate a score $f(x, x') = \phi(x) \cdot \phi(x')$.

4 Experimental Evaluation

Baselines. We evaluate the predictive performance of our approach against two different types of unsupervised baselines: scores computed on structural properties of the graph only and scores computed as a predefined distance measure on the feature vectors only. The first corresponds to the *topological* baselines from Liben-Nowell and Kleinberg in [13]; we will use here scores obtained by computing the common neighbors, Katz measure, Adamic/Adar measure and commute times in the graph. The second type of baselines are *distance-based* where we will predict the score from any node to any other node by computing either Euclidean distance or Cosine similarity between their feature vectors.

In order to assess the improvement with respect to the prediction accuracy that a random decision process would achieve on each one of the considered datasets, we also report results for a random predictor which simply selects edges in the test set at random, according to the edge prior distribution computed on the graph at time t: $p((x, x') \text{ linked}) = \frac{2 \cdot |E_t|}{|V_t| \cdot (|V_t| - 1)}$.

Datasets. As real dataset we will use the NIPS coauthorship network. Feature vectors of authors in this network are represented by the bag-of-words they use in the abstracts of their published papers. For separating the network in training G_t and test G_{t+1}, we used years from y to 2000 for training and years from y to 2003 for test, resulting in a network of n_t authors in G_t and n_{t+1} authors in G_{t+1} (values for n_t and n_{t+1} are reported in table 1). In order to have more diverse bag-of-words representation for collaborating authors, we discarded authors with less than two papers in the considered period. We created 3 such splitting of the NIPS dataset for years $y \in \{1998, 1996, 1992\}$ which we be named nips-1, nips-2 and nips-3. We also generated synthetic networks in order to test

[1] In the case of directed graphs one can instead consider the learning of an embedding for the concatenation of pairs of feature vectors

	Clust500	Clust750	FF500	FF750	nips-1	nips-2	nips-3
nodes in G_t	500	750	332	500	127	242	482
nodes in G_{t+1}	500	750	500	750	232	359	598
edges in G_t	2484	3583	1916	2832	379	802	1788
attributes	2	2	3	3	10770	12578	13473

Table 1: Dataset statistics

whether our approach was able to recover a prespecified distance measure embedded in the graph. We used two different types of graph generators. The first is a temporal evolution network based on the forest-fire generator from [12]. The idea of the basic forest-fire generator is that nodes arrive over time to the network and form out-links to some subset of earlier nodes, called its ambassadors; then, every new node starts to link to the predecessors of those ambassadors by "burning" links forward (i.e. adding an edge) with probability p_{forward} and "burning" links backwards with probability $p_{\text{backwards}}$. One can view such a process as intuitively corresponding to a model by which an author of a paper identifies references to include in the bibliography. The second type of synthetic generator we use is a cluster-structured network where an initial fixed set of nodes is divided in a set of communities and then, edges are drawn at random with within-community probability p_{in} and outside-community probability of p_{out}.

To create the dependency between feature vectors of the nodes and the network topology, we decide to bias the linking probabilities (that is, choosing an ambassador for the forest fire, or probablity p_{in} and p_{out} for the clustered-structured generator) in both generative processes by the Euclidian distance. Nodes in the network will be intially known to be separated by clusters; every cluster i produces a two dimensional feature vector for its nodes based on a Gaussian $\mathcal{N}(\mu_i, \sigma)$. Finally, linking probabilities in the generation process are biased proportional to the closeness in the Euclidian distance sense: the closer the feature vectors of the nodes in the Euclidian space, the higher will be the probability that they link to each other.

The datasets created from this synthetic process will be called here FF500 and FF750 for the forest-fire, and Clus500 and Clust750 for the clustered graph, with 500 and 750 nodes respectively. Each one of these datasets will be divided between training and test to the proportion of $1/3$. The test selected by forest-fire corresponds to 1/3 of the nodes arrived at the end in the network; the test selected by the clustered-graph corresponds to 1/3 of the known edges in the graph. The first corresponds to a temporal prediction problem, the second to a structural one.

Evaluation measure. Motivated by preference learning problems and because it is more sensitive in imbalanced datasets, our comparison for the link prediction scenario will be based on area under the ROC curve (AUC). This is the natural measure used also in previous related work for link prediction [18, 15, 14].

Results. The preliminary results of our approach are shown in Table 2.

	Clust500	Clust750	FF500	FF750	nips-1	nips-2	nips-3
Euclidean	**0.81**	**0.79**	**0.68**	**0.70**	0.34	0.27	0.20
Cosine	0.54	0.53	0.50	0.54	**0.48**	**0.39**	**0.32**
Common neigh	0.58	0.54	0.30	0.32	0.27	0.22	0.19
Katz	0.75	0.76	0.17	0.18	0.27	0.21	0.17
Adamic/Adar	0.58	0.54	0.30	0.32	0.27	0.22	0.19
Commute time	0.18	0.02	0.39	0.36	0.29	0.21	0.20
Random	0.40	0.38	0.33	0.33	0.27	0.22	0.19
Our approach	**0.81**	**0.79**	**0.69**	**0.71**	**0.50**	**0.40**	**0.31**

Table 2: Comparison of the different methods based on AUC. Bold numbers indicate best results.

We can notice from table 2 that the approaches relying only on the structure of the graph to compute the score, have performances close or worse than random for all datasets except Clust500, Clust750. This is because the link prediction setting in Clust is *structural* while in the other datasets it is *temporal*; these predicting approaches based on structure of the graph only cannot adapt to new

nodes that have not been observed during training time. We note that our approach performance for the synthetic datasets is very close to that of using only the Euclidean distance as ranking function. This can be explained by the fact that the dependencies in those datasets are fixed by construction by the Euclidean distances between the feature vectors associated with the nodes. On the other hand on the nips datasets, where the nodes are represented by bag-of-words the performance is closer to the one of using only the Cosine similarity measure between the nodes to rank their affinities. Experiments on a higher scale need to be performed in order to see if our approach can improve with respect to approaches using only the feature space. We can however note that our approach seems to learn each time the correct distance.

5 Conclusions

This paper deals with the problem of link prediction in graphs. Our approach is based on learning a preference score between pairs of nodes, indicating their (mutual) affinity given the current network configuration and the content of the nodes. We proposed a modelisation of the link prediction problem as a learning to rank link preferences based on the features of the nodes with the partial feedback of the structure (edges and shortest paths) observed in the graph at training time. As in previous related work, we assume there is a latent space in which the features of the nodes reside, and links are formed based on the unknown distances between nodes in this latent space. We propose therefore a way to learn this distance function via supervision of the structure observed in the graph. The advantage of our approach over current solutions is that the link prediction problem becomes natural also in the temporal setting scenario, where new nodes might be arriving in the test time. Preliminary experiments are encouraging: our algorithm is able to learn the correct distance function in synthetic networks and it outperforms several baselines in real networks.

References

[1] Evrim Acar, Daniel M. Dunlavy, and Tamara G. Kolda. Link prediction on evolving data using matrix and tensor factorizations. In *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*, ICDMW '09, pages 262–269, 2009.

[2] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah. Signature Verification using a Siamese Time Delay Neural Network. In *Advances in Neural Information Processing Systems 6*, 1993.

[3] C.J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G.N. Hullender. Learning to rank using gradient descent. In *ICML*, pages 89–96, 2005.

[4] S. Chopra, R. Hadsell, and Y. LeCun. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *Proc. of Computer Vision and Pattern Recognition Conference*, 2005.

[5] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning: An introduction. In Johannes Fürnkranz and Eyke Hüllermeier, editors, *Preference Learning*, pages 1–17. Springer-Verlag, 2010.

[6] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning and ranking by pairwise comparison. In Johannes Fürnkranz and Eyke Hüllermeier, editors, *Preference Learning*, pages 65–82. Springer-Verlag, 2010.

[7] D. Grangier and S. Bengio. Exploiting hyperlinks to learn a retrieval model. In *NIPS Workshop on Learning to Rank*, 2005.

[8] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, March 1953.

[9] Myunghwan Kim and Jure Leskovec. The network completion problem: Inferring missing nodes and edges in networks. In *SDM*, pages 47–58, 2011.

[10] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42:30–37, August 2009.

[11] Jérôme Kunegis and Andreas Lommatzsch. Learning spectral graph transformations for link prediction. In *ICML*, page 71, 2009.

[12] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 177–187, 2005.

[13] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 58:1019–1031, 2007.

[14] Ryan N. Lichtenwalter, Jake T. Lussier, and Nitesh V. Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 243–252, 2010.

[15] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *ECML/PKDD (2)*, pages 437–452, 2011.

[16] Purnamrita Sarkar, Deepayan Chakrabarti, and Andrew W. Moore. Theoretical justification of popular link prediction heuristics. In *IJCAI*, pages 2722–2727, 2011.

[17] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems 16*. Cambridge, MA, 2004.

[18] Shenghuo Zhu, Kai Yu, Yun Chi, and Yihong Gong. Combining content and link for classification using matrix factorization. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 487–494, 2007.